Announcements

- A5 due tonight
- A6 is out, remember to get started early
- For the next lecture, you MUST watch the tutorial on the shortest path algorithm beforehand: http://www.cs.cornell.edu/courses/cs2110/2017f/online/shortestPath/shortestPath.html
- The class on 4/10 will assume that you understand it. Watch the tutorial once or twice and execute the algorithm on a small graph.
- Complete Quiz 4 by 4/9

Graphs

Representing Graphs

Graph Algorithms

Search on Graphs
**Depth-First Search**

Intuition: Recursively visit all vertices that are reachable along unvisited paths.

```java
class Node {
    boolean visited;
    List<Node> neighbors;
}

//** Visit all nodes reachable on unvisited paths from u.
// Precondition: u is unvisited. */
public static void dfs(int u) {
    visit(u);
    for all edges (u,v):
        if(!visited[v]):
            dfs(v);
}
```

DFS Quiz

- In what order would a DFS visit the vertices of this graph? Break ties by visiting the lower-numbered vertex first.
  - 1, 2, 3, 4, 5, 6, 7, 8
  - 1, 2, 5, 6, 3, 6, 4, 7, 8
  - 1, 2, 5, 6, 4, 7, 8
  - 1, 2, 5, 3, 6, 7, 4, 8

**Depth-First Search in Java**

```java
public class Node {
    boolean visited;
    List<Node> neighbors;
}

//** Visit all nodes reachable on unvisited paths from this node.
// Precondition: this node is unvisited. */
public void dfs() {
    visited = true;
    for each edge (u,v):
        if (!v.visited) { v.dfs(); }
}
```

Intuition: Iteratively process the graph in "layers" moving further away from the source node.

```java
public class Node {
    boolean visited;
    List<Node> neighbors;
}

//** Visit all nodes reachable on unvisited paths from u.
// Precondition: u is unvisited. */
public static void bfs(int u) {
    Queue q = new Queue();
    q.add(u);
    while (q is not empty) {
        u = q.remove();
        if (u not visited) {
            visit u;
            for each (u, v):
                if (!v.visited) { v.dfs(); }
        }
    }
}
```

Breadth-First Search

Intuition: Recursively visit all vertices that are reachable along unvisited paths.

```java
public static void dfs(int u) {
    visit(u);
    for all edges (u,v):
        if(!visited[v]):
            dfs(v);
}
```

Intuition: Recursively visit all vertices that are reachable along unvisited paths.

```java
public static void dfs(int u) {
    visit(u);
    for all edges (u,v):
        if(!visited[v]):
            dfs(v);
}
```

Intuition: Recursively visit all vertices that are reachable along unvisited paths.

Worst-case running time? \(O(n + m)\)

Worst-case space? \(O(n)\)

Suppose there are \(n\) vertices that are reachable along unvisited paths and \(m\) edges:
Analyzing BFS

```
/** Visit all nodes reachable on unvisited paths from u. */
public static void bfs(int u) {
    Queue q = (u); // Not Java!
    while (!q.empty()) {
        u = q.remove();
        if (u not visited) {
            visit u;
            for each (u, v):
                q.add(v);
        }
    }
}
```

Intuition: Iteratively process the graph in "layers" moving further away from the source node.

Suppose there are \( n \) vertices that are reachable along unvisited paths and \( m \) edges:

- Worst-case running time: \( O(n + m) \)
- Worst-case space: \( O(m) \)

BFS Quiz

- In what order would a BFS visit the vertices of this graph? Break ties by visiting the lower-numbered vertex first.
  - 1, 2, 3, 4, 5, 6, 7, 8
  - 1, 2, 3, 4, 5, 6, 7, 7, 8
  - 1, 2, 5, 3, 6, 4, 7, 8
  - 1, 2, 5, 6, 3, 7, 4, 8

Comparing Search Algorithms

- **DFS**
  - Visits: 1, 2, 3, 5, 7, 8
  - Time: \( O(n + m) \)
  - Space: \( O(n) \)

- **BFS**
  - Visits: 1, 2, 5, 7, 3, 8
  - Time: \( O(n + m) \)
  - Space: \( O(m) \)